CS 106
Introduction to Computer Science I

07 / 08 / 2021

Instructor: Michael Eckmann

Today's Topics

- Questions / comments?
- Nested if statements
- Logical operators (and, or, not)
- Reserved words and valid variable names
- random numbers
- while loop

Nested ifs

- If statements can have other if statements nested within them.
- Pay attention to the indentation to tell which if the elif or else connect to.
- Let's look at a couple of examples.

Logical operators

- **and or not** are three operators that use bools as operands and result in a bool

- Truth table for **and**

False **and** False results in False
False **and** True results in False
True **and** False results in False
True **and** True results in True

Logical operators

- **and or not** are three operators that use bools as operands and result in a bool

- Truth table for **or**

False **or** False results in False
False **or** True results in True
True **or** False results in True
True **or** True results in True

Logical operators

- **and or not** are three operators that use bools as operands and result in a bool

- Truth table for **not**

not True results in False

not False results in True

Logical operators

- Let's write some ifs using and and or

Let's look at the reserved words list

- Python has a list of reserved words which already have meaning and are not allowed to be used as variable names.
- We've used some of them already: True, False, if, elif, else, and, or, not
- Let's look at the complete list
- Variable names cannot be among the reserved words, and are only allowed to contain letters, digits, _ and they cannot start with a digit.

Random number generation

- To use random numbers in python we must import the random module like so:

```
import random
```

- Then we can call any of the functions defined in the random module by preceding their name with random. (random followed by a dot)
- e.g.

```
random.random()
# returns a random number in [0,1.0)
```

Random number generation

```
random.random()
# returns a random number in [0,1.0)
```

- That is, it will be a float in the range 0 to 1 (not including 1)
- Let's run it a few times and see what we get

Random number generation

```
random.randint(1,5)
# returns a random int among 1,2,3,4,or 5
```

- The first parameter of randint is the low end of the range (doesn't have to be 1)
- The second parameter of randint is the high end of the range (doesn't have to be 5)
- The function returns a random int in that range (inclusive on both ends)

While loop

- A while loop can be used to repeat lines of code as long as some condition (boolean expression) remains true.
- Syntax is

```
while ____:
    # line 1 of body of the loop
    # line 2 of body of the loop
    # …
    # last line of body of the loop
```

- ____ is a boolean expression (evaluates to True or False)

While loop

- Let's read and try to predict what this loop does.

```
count = 0
while count < 10:
    print(count)
    count += 1
```

While loop

- Let's read and try to predict what this loop does.

```
count = 0
while count > 10:
    print(count)
    count += 1
```

While loop

- Let's read and try to predict what this loop does.

```
count = 0
while count >= 0:
    print(count)
    count += 1
```

While loop

- Let's write a guessing game using
  - User input
    - We'll ask for the range they wish to guess among
    - We'll ask for their guess
  - Random number generation
  - While loop to allow repeated guesses if incorrect
  - Modify it to tell the user higher/lower